

## **REMARKS**

Applicant respectfully requests reconsideration of this application in view of the foregoing amendments and the following remarks.

### **Rejections Under 35 U.S.C. § 102**

The Office rejected Claims 1-4, 6-14, 16-32 and 34-37 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,044,388 (hereinafter "DeBellis"). Applicant respectfully traverses the rejection.

### **Claims 1, 11, 25 and 32**

*Claims 1, 11, 25 and 32 as amended, recite:*

1. A method comprising:  
collecting entropy data, wherein the entropy data includes central processing unit data and operating system data;  
storing the entropy data in a nonvolatile memory;  
updating the entropy data stored in the nonvolatile memory with newly collected entropy data; and  
generating a string of random bits from the entropy data stored in the nonvolatile memory.

11. One or more computer-readable memories containing a computer program that is executable by one or more processors, the computer program causing the one or more processors to:  
collect entropy data, wherein the entropy data includes central processor unit data and operating system data;  
store the entropy data in a nonvolatile memory;  
update the entropy data stored in the nonvolatile memory with newly collected entropy data; and  
generate a string of random bits from the entropy data stored in the nonvolatile memory.

25. An apparatus comprising:  
a nonvolatile memory configured to store entropy data, wherein the entropy data stored in the nonvolatile memory is updated regularly by hashing the entropy data stored in the nonvolatile memory with newly collected entropy data; and  
a random number generator, coupled to the nonvolatile memory, to generate strings of random bits using the entropy data received from the nonvolatile memory, wherein the entropy data is collected from a central processing unit and an operating system executed by the central processing unit.

32. One or more computer-readable media having stored thereon a computer program that, when executed by one or more processors, causes the one or more processors to:  
collect entropy data from the one or more processors and one or more operating systems executed by the one or more processors;  
store the collected entropy data in a nonvolatile memory;  
update the entropy data stored in the nonvolatile memory with newly collected entropy data; and  
produce a string of random bits from the entropy data stored in the nonvolatile memory.

DeBellis (U.S. 6,044,388)

DeBellis discloses that a computer system 100 includes a pseudorandom number generator 102 that is part of a cryptographic module 104. (Column 6, lines 32-37, Figure 1) The cryptographic module 104 generates pseudorandom numbers...by combining a time dependent value "T" with a secret value "S" and passing the result through a hash function to generate a random number. (Column 4, line 67 – Column 5, line 5). The time dependent value "T" is generated by a real time counter and is incremented at a sufficiently rapid rate that an adversary cannot predict the exact value of T. (Column 6, Lines 58-64) Secret value "S" is a

128 bit randomized state which is updated by two feedback functions (F1 and F2). Feedback function "F1" updates "S" whenever the cryptographic module is idle (*i.e.* not performing other activities). (Column 7, lines 7-10) Feedback function "F2" updates "S" when an external random event occurs (*i.e.* a command from an application program 110). (Column 7, Lines 17-25).

Applicant submits that DeBellis fails to disclose the elements of Claims 1-4, 6-14, 16-32, and 34-37. Specifically, Applicant submits that DeBellis fails to disclose "collecting entropy data, wherein the entropy data includes central processor data and operating system data", as recited in Claim 1. (Emphasis added.) DeBellis also fails to disclose or suggest the features of Claims 11, 25 and 32 as shown above.

DeBellis describes that pseudorandom numbers are generated in a cryptographic module... by concatenating time dependent value "T" with a secret value "S". (Column 4, line 67 – Column 5, line 5) DeBellis goes on to say that cryptographic module 104 may include other cryptographic functions for encryption, key management, digital signature processing and the like. (Column 6, lines 37-40) Since T and S (entropy data) reside in a cryptographic module 104, they do not include operating system data. (Emphasis Added)

Moreover, time-dependent value "T" is generated by a real-time counter (Column 6, lines 58-63) and secret value "S" is updated by feedback function "F1" when the cryptographic module is idle and feedback function "F2" when an

external event occurs (i.e. command from an application 110). (Column 7, lines 7-21) (Emphasis added) Since T is generated by a counter and S is updated when the cryptographic module is idle or when an external event occurs, they do not include central processing unit data. (Emphasis Added)

Thus, DeBellis does not disclose “collecting entropy data, wherein the entropy data includes central processor data and operating system data,” as recited in independent Claim 1. (Emphasis added.) Accordingly, Applicant respectfully submits that Claim 1 is allowable over DeBellis. Claims 11, 25, and 32 are also allowable for reasons similar to those discussed with respect to Claim 1.

**Claims 4, 6-10, 27, 28, 30, 31, 34, 36 and 37**

Claims 4, 6-10, 27, 28, 30, 31, 34, 36 and 37 depend from independent Claims 1, 25 or 32 and are allowable at least due to their dependency from Claims 1, 25 or 32.

**Claims 12, 18, 19 and 24**

*Claims 12, 18, 19 and 24, as amended, recite:*

12. A method comprising:
  - receiving a request for a random number;
  - retrieving, from a protected portion of an operating system kernel, entropy data that is regularly updated with newly collected entropy data, wherein the entropy data includes central processor unit data and an operating system data;
  - hashing the entropy data to create random seed data;
  - generating a string of random bits from the random seed data;
  - and

communicating the string of random bits to the requester of the random number.

18. One or more computer-readable memories containing a computer program that is executable by one or more processors, the computer program causing the one or more processors to:

receive a request for a random number;

retrieve entropy data from a protected portion of an operating system kernel, wherein the entropy data includes central processor unit data and an operating system data;

hash the entropy data to create random seed data;

generate a string of random bits from the random seed data;

and

communicate the string of random bits to the requester of the random number.

19. A method comprising:

collecting entropy data, wherein the entropy data includes central processing unit data and operating system data;

storing the entropy data in a protected portion of an operating system kernel; and

generating a string of random bits based on the entropy data.

24. One or more computer-readable memories containing a computer program that is executable by one or more processors, the computer program causing the one or more processors to:

collect entropy data from a central processing unit and an operating system executed by the central processing unit;

store the entropy data in a protected portion of an operating system kernel; and

generate a string of random bits based on the entropy data.

First, DeBellis does not disclose retrieving from a protected portion of an operating system kernel, entropy data, as recited in Claim 12. DeBellis also lacks features of each of Claims 18, 19 and 24. (Emphasis added) Rather, DeBellis describes that “registers 254 (stores time dependent value T) and 256 (stores secret value S) are implemented in nonvolatile storage...” (Column 9, lines 15-19)

Although DeBellis does not define the term “register”, most computer architectures move data from a main memory into “data registers”, perform an operation on the data and then move the result back into main memory. Data registers are typically implemented as register files in a central processing unit (CPU), but they have also been implemented using flip-flops, high speed core memory, and thin film memory.

In contrast, as described in one implementation, the Applicant retrieves [or stores] entropy data from a protected portion of the operating system kernel (i.e. a portion of the operating system kernel that is not accessible by an application program).” (Page 7, lines 8 – 10) It is well known that an “operating system kernel” is the central component of most computer operating systems, which connects the application software to the computer hardware (e.g. memory, processors and I/O devices). Thus, DeBellis’ teaches that *T and S (entropy data) are stored in register files in the CPU*, while the Applicant stores entropy data in the *computers operating system*.

Therefore, DeBellis fails to disclose *retrieving from a protected portion of an operating system kernel, entropy data* (Claims 12, 18, 19 and 24) (Emphasis added)

Second, as previously noted, DeBellis does not disclose *collecting entropy data, wherein the entropy data includes central processor data and operating system data*, as recited in Claims 12, 18, 19, and 24. (Emphasis added)

For these reasons, Applicant respectfully submits that Claims 12, 18, 19 and 24 are allowable over DeBellis.

**Claims 14, 16, 17, 21, 22 and 23**

Claims 14, 16, 17, 21, 22, and 23 depend from independent Claims 12 and 19 and are allowable at least due to their dependency from Claims 12 and 19.

**Claim 7**

With reference to Claim 7, the Office stated that [DeBellis] calls for entropy data to not be readily ascertainable by an attacker. (Column 7, lines 19-25) (Emphasis added)

*Claim 7 recites:*

7. A method as recited in claim 1 wherein the method is executing on a system and the entropy data is inaccessible by an application program executing on the system.

First, as previously noted Claim 7 depends from independent Claim 1 and is allowable at least due to its dependency from Claim 1.

Second, Claim 7 is also allowable because DeBellis fails to disclose the entropy data is inaccessible by an application program executing on the system. (Emphasis added)

DeBellis discloses that feedback function F2...is invoked by an external input, specifically an external randomize event (e) ... [The] external randomized event (e) maybe the receipt of a command from an application [program] 110... (Column 7, lines 17-20) DeBellis further states that, "Other external events and parameters could be used instead; the important consideration is that they are not readily ascertainable by an attacker." (Column 7, lines 23-25) Thus, DeBellis discloses that feedback function F2 could be "invoked" by a command from an application program, so long as it is not readily ascertainable by an attacker.

(Emphasis added) In contrast, Applicant teaches that entropy data is "inaccessible" by an application program executing on the system. Therefore, DeBellis discloses that feedback function F2 maybe invoked by a command from an application program while the Applicant teaches that entropy data is inaccessible by an application program executing on the system. (Emphasis added)

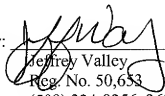
Therefore, DeBellis does not disclose that entropy data is inaccessible by an application program executing on the system.. (Emphasis added) For these reasons, Applicant respectfully submits that Claim 7 is allowable over DeBellis.

### **Conclusion**

Applicant respectfully submits that Claims 1, 4, 6-12, 14, 16-19, 21-25, 27, 28, 30-32, 34, and 36-37 are in condition for allowance. Applicant respectfully requests reconsideration and issuance of the subject application. Should any matter in this case remain unresolved, the undersigned respectfully requests a telephone conference with the Examiner to resolve any such outstanding matter.

Respectfully Submitted,

Date: May 4 2007

By:   
Jeffrey Valley  
Reg. No. 50,633  
(509) 324-9256x262